Basic Block Tools for Mixed Mode Binaries

Hoi Vo

Binary Technologies
PPRC
Microsoft Research

Jan 08 2004

Agenda

- Introduction
- Background On BBT
- The Current Process
- Mixed Mode Twist
- Instrumentation
- Layout Optimization
- The Future With Phoenix

Binary Technologies (BIT)

- Improve performance for Windows Platform
  - With focus on managed code
  - Better profile gathering tools without static instrumentation
- Enable solutions that take advantage of both Static and Runtime Analysis
- Better control of compilation process from IL -> code generation -> runtime
  - Back-mapping between different code transformation phases
  - Flexible API to control all phases of compilation

Introduction

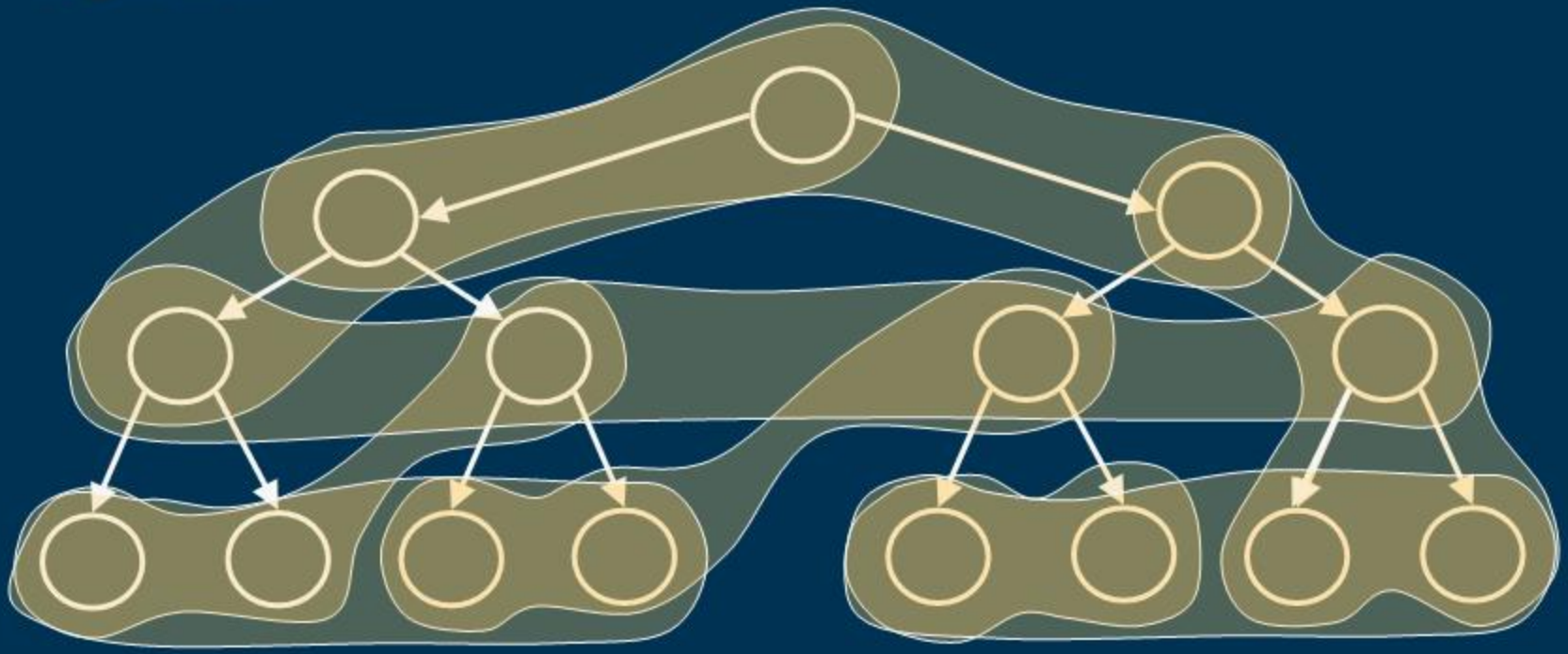
- BBT is a post-link optimization toolset
- Binary instrumentation
- Profile-guided transformation
  - Use static analysis coupling with dynamic profile data to detect opportunities
  - Automatic binary reordering at basic block level
  - Handle both static code and data blocks
- Improve memory hierarchy performance
  - Minimize memory latencies

Code Optimizations

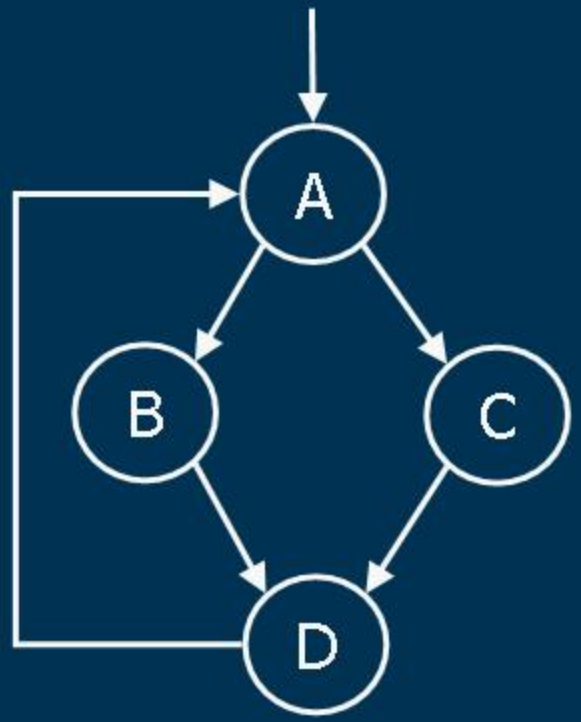
- Code placement
  - Basic block clustering based on temporal profile data
- Classic code optimizations
  - Control flow restructuring
  - Code specialization
  - Partial inlining

Basic Block Layout

- Co-locate basic blocks that need to be in memory together

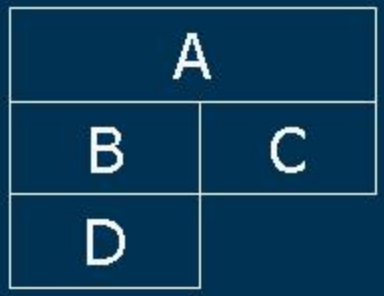


Control Flow Restructuring

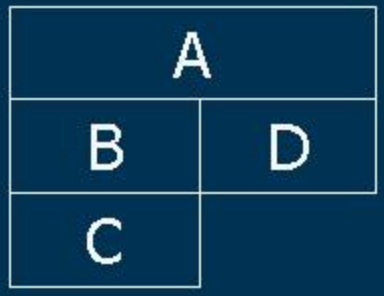


Control Flow Trace:

A-B-D	10
A-C-D	1



Org

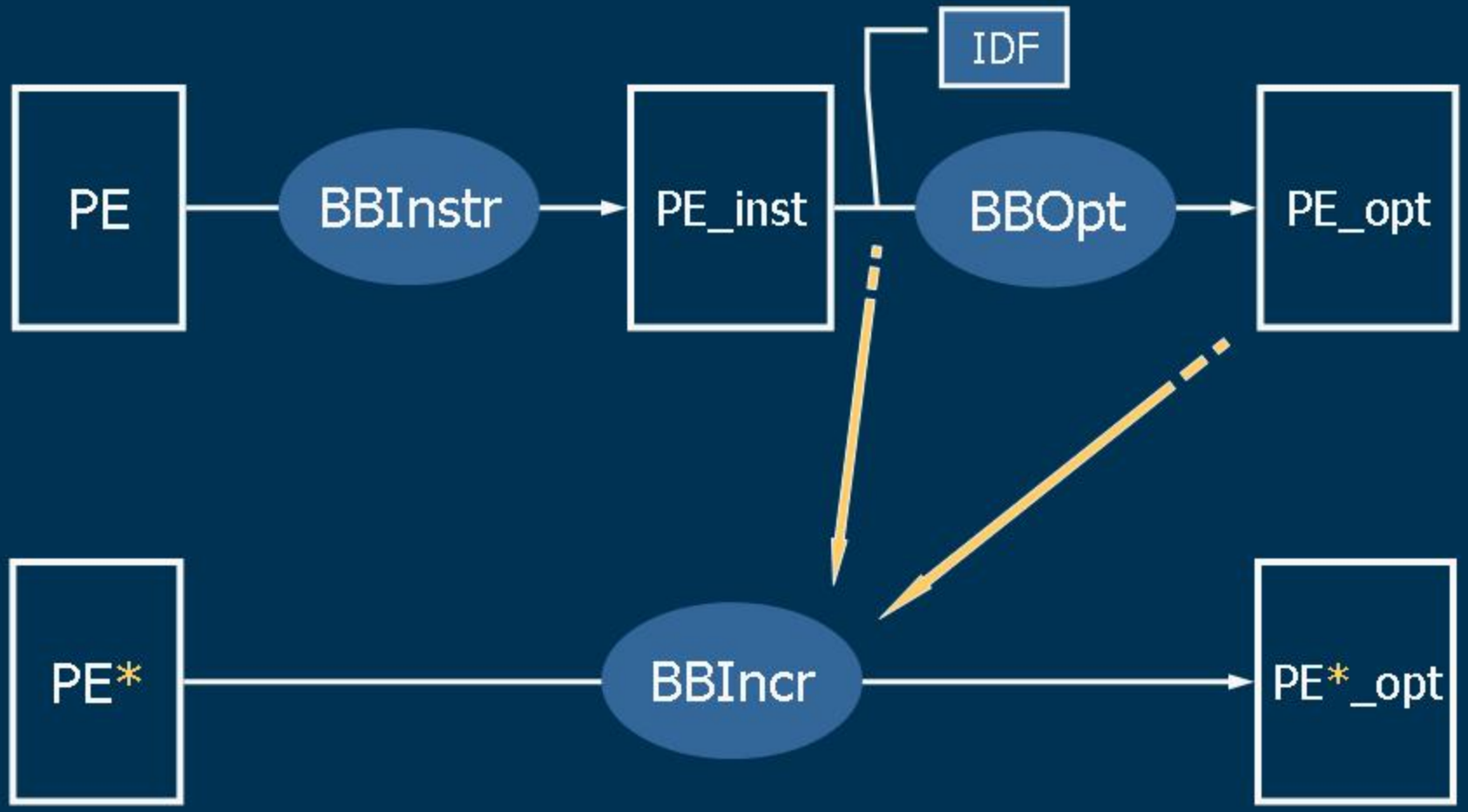


Opt

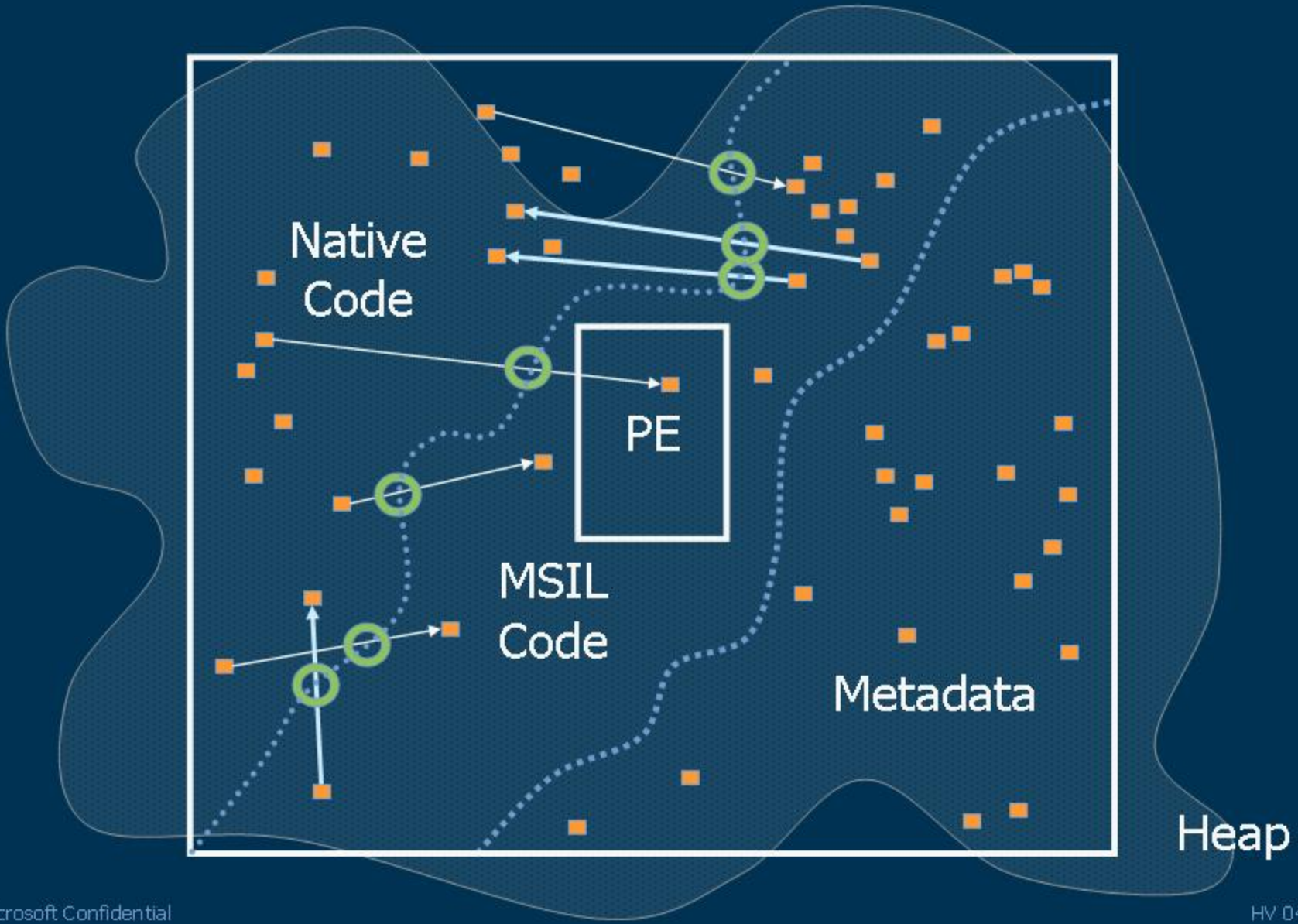
Data Optimizations

- Data optimization
  - Data layout (based on code references and profiles)
- Static data
  - Co-locate with code blocks that reference them
- Resources
  - Co-locate resources that are being loaded around the same time marker

The Current BBT Process



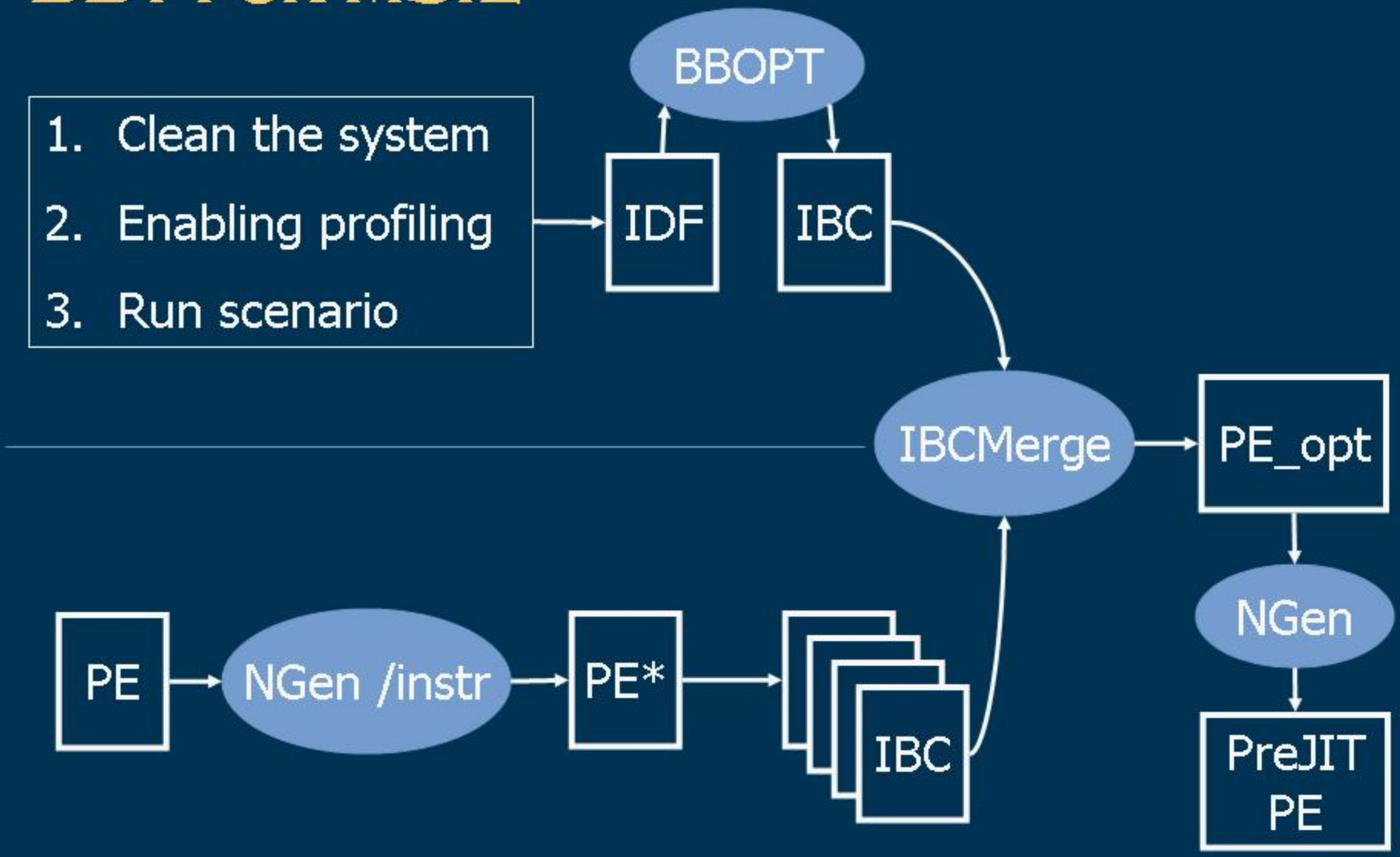
Mixed Mode Binaries



Challenges With Mixed Mode

- Instrumentation
  - Control flow between Nat and Mgd code
  - Metadata accesses
- Optimization
  - Native (Nat) and Managed (Mgd) code mixed together
  - Mgd methods cannot be split at the IL level
  - Metadata reordering
- Incremental
  - Diffing Mgd code more complex
  - Cannot deal with methods that switch from being Nat to Mgd or vice-versa

BBT FOR MSIL

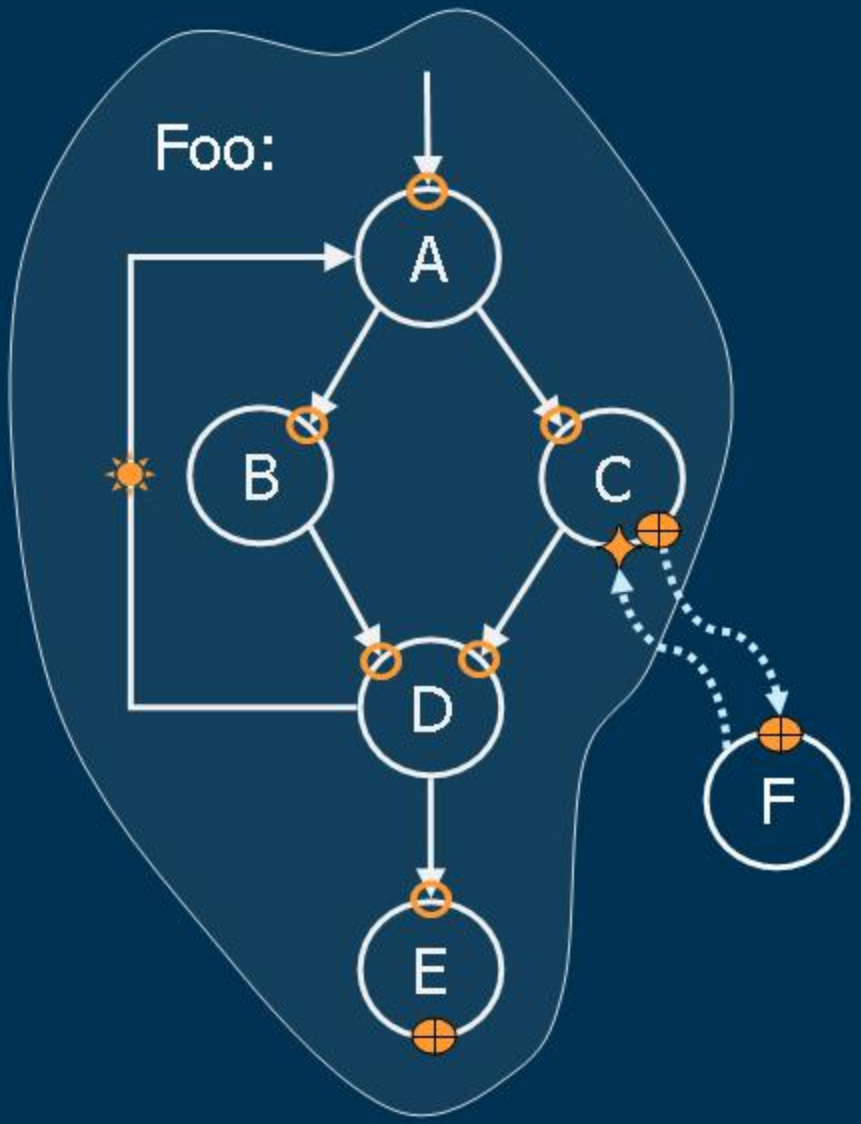

1. Clean the system
2. Enabling profiling
3. Run scenario
BBOPT
IDF
IBC
IBCMerge
PE_opt
NGen
PreJIT PE
PE
NGen /instr
PE*
IBC

Current State

- As smooth as riding a mechanical bull at top speed
- Too many steps to err during the process
- Current optimization impact is limited to IL images
- No common methodologies for validation
- Team needs are too different

So where should we start?

Mixed Mode Instrumentation



Native Code:

- Temporal
- Block count
- Edge flow
- Indirect flow

Managed Code:

- Temporal
- Block count

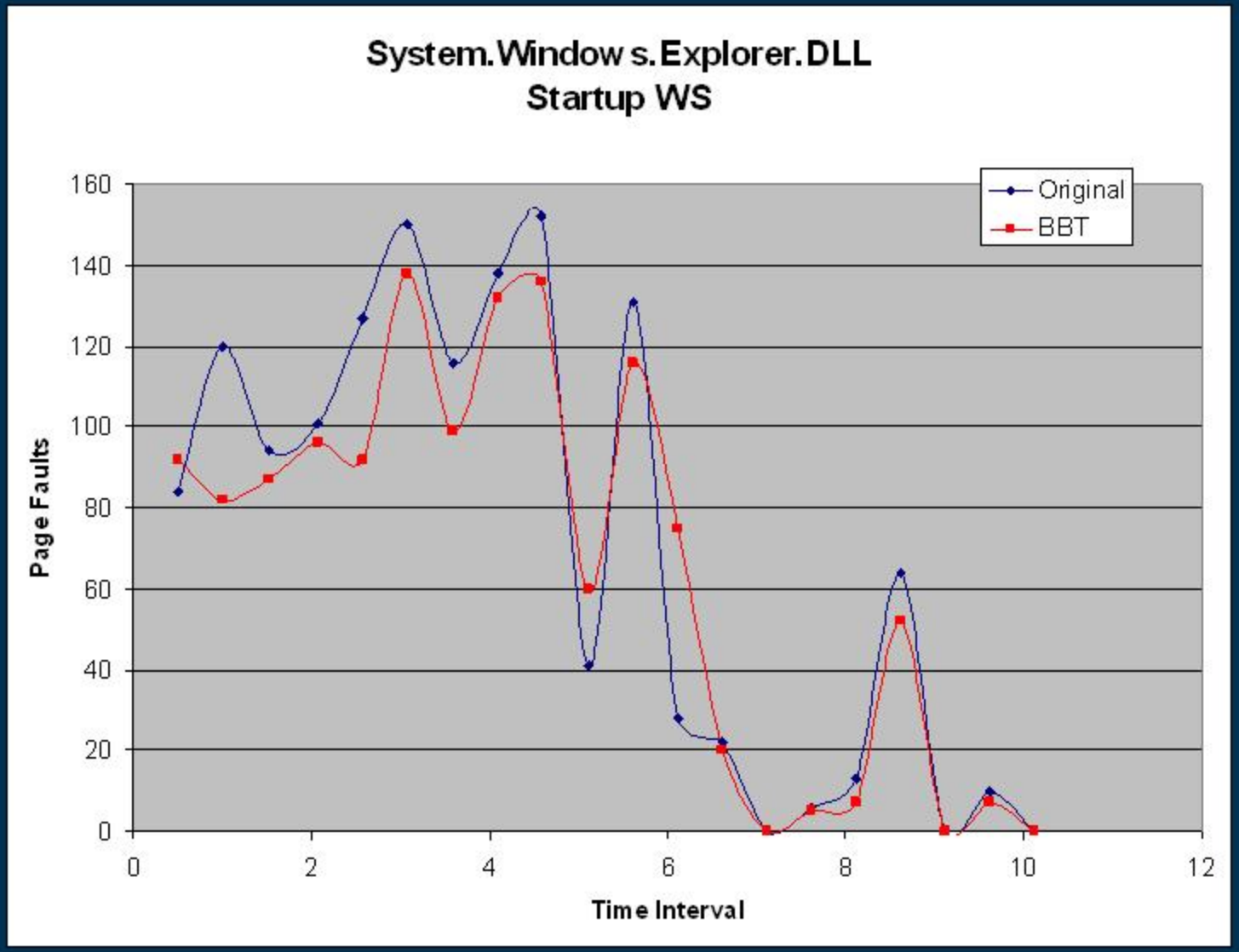
Mixed Mode Optimization

- Native code
  - No change – basic block reordering
- Managed code
  - Re-ordered at the function or method granularity
  - Produce IBC for NGen
- Native <--> Managed transitions
  - __mep_xxx [Managed Entry Points]
  - Light PInvoke [Native Transition Points]

BBIncr With Mixed Mode

- No change
- Requires
  - Old Org binary + PDB
  - Old IDF + Old KEY
  - New Org binary + PDB
- Or
  - Old Opt binary + PDB
  - New Org binary + PDB

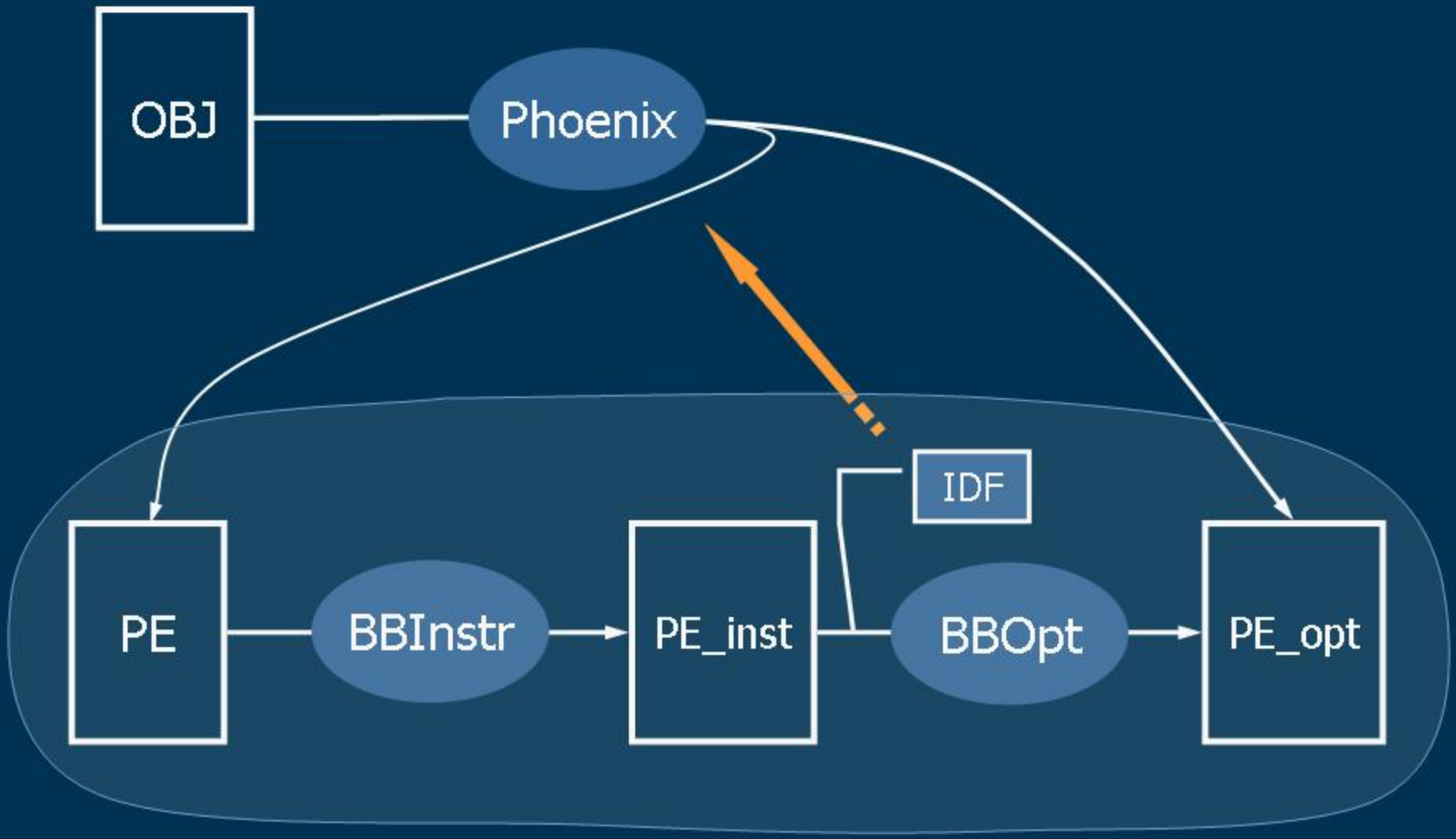
Preliminary Results



Future Directions For BBT

- Instrumentation
  - Reduce dependency on static instrumentation
  - Can leverage profile data from any sources
  - Any properties that could be leveraged for optimization
- Optimization
  - Certain optimizations will be done within compiler codegen engine
  - Layout engine must address all code and all data
  - Can leverage stale profile data
- Adaptive mode for dynamic compilation model (JIT)

Phoenix & BBT



Resources – Q&A

- http://bbt
- Aliases: BBTNews; BiTInfo